1a- Aim: simple Calculator

Code:

```
#include<iostream>
using namespace std;
int main(){
     int a,b,c;
     char op;
    cout<<"Enter First Number="<<endl;
    cin>>a;
    cout<<"Enter the operator Number="<<endl;
    cin>>op;
    cout<<"Enter Second Number="<<endl;
    cin>>b;
    if(op=='+')
{
    c=a+b;
    cout<<"Addition of two number is:"<<c;
}
else if(op=='-'){
        c=a-b;
    cout<<"Substraction of two number is:"<<c;
}
else if(op=='*'){
    c=a*b;
    cout<<"Multiplication of two number is:"<<c;
}
else if(op=='/'){
    c=a/b;
    cout<<"Division of two number is:"<<c;
}
else{
   cout<<"/n Invalide Operator.Enter a Valid Opeartor";
}
}
```

1B-Aim: convert into hours,minutes, and seconds.

Code:

```
#include<iostream>
using namespace std;
int main(){
int sec,hr,min;
cout<<"Program to convert Seconds into Hours,Minutes and Seconds"<<endl;
cout<<"Enter the value of Seconds:"<<endl;
cin>>sec;
min=sec/60;
hr=min/60;
cout<<sec<<"Seconds is equivalent to"<<int(hr)<<"Hours";
cout<<int(min%60)<<"Minutes"<<int(sec%60)<<"Seconds";
}
```

2a- Aim : Largest number amongst them.

Code:

```
#include<iostream>
using namespace std;
int main(){
int n1,n2,n3;
cout<<"Enter First Number:";
cin>>n1;
cout<<"Enter Second Number:";
cin>>n2;
cout<<"Enter Third Number:";
cin>>n3;
if(n1>=n2 && n1>=n3)
```

```
   cout<<"Largest number:"<<n1;
else if(n2>=n1 && n2>=n3)
cout<<"Largest number:"<<n2;
else if(n1==n2||n1==n3||n2==n3)
cout<<"All the number are equal";
else
   cout<<"Largest number:"<<n3;
}
```

2a- Aim : Largest number amongst them.

Code:

```
#include<iostream>
using namespace std;
class student{
private:
   char name[20];
   int RN;
public:
   int getdetails(void);
   int putdetails(void);
};
int student::getdetails()
{
   cout<<"Enter the Name of Students:";
   cin>>name;
   cout<<"Enter the Roll Number of student:";
   cin>>RN;
}
int student::putdetails(){
cout<<"\n Displaying the details of the student:";
cout<<"\n Name:"<<name<<endl;
cout<<"Roll Number:"<<RN;
}
int main(){
student s;
s.getdetails();
s.putdetails();
}
```

4b- Aim : friend function fot distance

Code:

```
#include<iostream>
using namespace std;

class Distance {
private:
   int ft;
   int inch;

public:
   Distance(int f = 0, int i = 0) : ft(f), inch(i) {}

   friend void addDistance(Distance d1, Distance d2, Distance& sum);

   void display() {
      cout << ft << "ft " << inch << "inch" << endl;
   }
};

void addDistance(Distance d1, Distance d2, Distance& sum) {
   sum.inch = d1.inch + d2.inch;
   sum.ft = d1.ft + d2.ft + (sum.inch / 12);
```

```
    sum.inch = sum.inch % 12;
}

int main() {
    Distance dist1(5, 9);
    Distance dist2(6, 11);
    Distance sum;
    addDistance(dist1, dist2, sum);

    cout << "Sum of the distance: ";
    sum.display();

    return 0;
}
```

5b- Aim : Static Demo

Code:

```
#include<iostream>
using namespace std;
class StaticDemo
{
    private:
    static int count;
    public:
    StaticDemo()
    {
        count ++;
    }
    static void showCount()
    {
        cout<<"Count:"<<count<<endl;
```

```
   }
};
int StaticDemo::count=0;
int main(){
   StaticDemo obj1,obj2,obj3;
   StaticDemo::showCount();
   return 0;
}
```

5b- Aim : Fibonacci series

Code:

```
#include<iostream>
using namespace std;
class Fibonacci{
   private:
         int n;
   public:
      Fibonacci(int num)
      {
      n=num;
      }
void generate()
{
   int first=0,second=1,next;
   cout<<"Fibonacci series:";
for(int i=0;i<n;i++)
   {
      cout<< first<<" ";
      next=first+second;
      first=second;
      second=next;
```

```
    }
    cout<<endl;
}
};
int main(){
int num;
cout<<"Enter the number in term:";
cin>>num;
Fibonacci fib(num);
fib.generate();
return 0;
}
```

6B Aim: Write a program in C++ to overload the operator uniary (-) for demonstrating operator overloading.

Code:

```
#include<iostream>
using namespace std;
class Num{
    private:
    int value;
    public:
    Num(int v):value(v)
    {}
    int getValue()const{return value;}
    Num operator - (){
        return Num (- value);
    }
};
int main(){
    Num n(5);
    cout<<"Original Num:"<<n.getValue()<<endl;
    Num negNum=-n;
    cout<<"after applying unary minus:"<<negNum.getValue()<<endl;
    return 0;

}
```

7A Aim: Write a program in C++ to implement the concept of method overriding along with a virtual function.

Code:

```
#include<iostream>
using namespace std;
class Oride
{
    public:
    virtual void calc()
    {
        cout<<"Calc is Multiplying the Number:"<<86*9<<endl;
    }
};
class Sum:public Oride
{
    public:
    void calc() override
    {
        cout<<"Calc is Adding the number:"<<20+5<<endl;
    }
};
class Remain:public Oride
{
    public:
    void calc()override
{
    cout<<"Calc is finding Remainder:"<<958%3<<endl;
}
};
int main()
{
    Oride* oride1;
    Sum S;
    Remain R;
    oride1=&S;
    oride1->calc();
    oride1=&R;
    oride1->calc();
    return 0;
}
```